Samuel A. Mason, Faiz Sayyid, Paul D. W. Kirk, Colin Starr, and David L. Wild*

MDI-GPU: Accelerating integrative modelling for genomic-scale data using GP-GPU computing

Abstract: The integration of multi-dimensional datasets remains a key challenge in systems biology and genomic medicine. Modern high-throughput technologies generate a broad array of different data types, providing distinct – but often complementary – information. However, the large amount of data adds burden to any inference task. Flexible Bayesian methods may reduce the necessity for strong modelling assumptions, but can also increase the computational burden. We present an improved implementation of a Bayesian correlated clustering algorithm, that permits integrated clustering to be routinely performed across multiple datasets, each with tens of thousands of items. By exploiting GPU based computation, we are able to improve runtime performance of the algorithm by almost four orders of magnitude. This permits analysis across genomic-scale data sets, greatly expanding the range of applications over those originally possible. MDI is available here: http://www2.warwick.ac.uk/fac/sci/systemsbiology/research/software/

1 Introduction

The integration of multi-dimensional datasets remains a key challenge in systems biology and genomic medicine. Modern high-throughput technologies generate a broad array of different data types, providing distinct — yet often complementary — information. The Multiple-Dataset Integration (Kirk et al., 2012) (MDI) algorithm permits the simultaneous clustering of an arbitrary number of datasets in a context dependent manner. This, and related, methods, show promise in recovering biologically meaningful clusters, as demonstrated in a number of recent studies (Barash and Friedman, 2002; Liu et al., 2006, 2007; Savage et al., 2010; Rogers et al., 2009; Savage et al.,

Samuel A. Mason, David L. Wild: Systems Biology Centre, University of Warwick, Coventry, CV4 7AL, UK
Faiz Sayyid: Department of Computer Science, University of Warwick, Coventry, CV4 7AL, UK
Paul D. W. Kirk, Colin Starr: MRC Biostatistics Unit, Cambridge Institute of Public Health, Cambridge, CB2 0SR, UK

2013; Yuan et al., 2011). By performing simultaneous clustering over multiple of complementary datasets our method is able to exploit correlations in clustering structure between datasets.

MDI is motivated by the desire to separate the *items* within *datasets* into statistically distinct clusters while exploiting any latent structure in cluster allocations across datasets. A flexible Bayesian mixture modelling approach is taken in order to allow the data to speak for themselves, and to avoid overly strong modelling assumptions. MDI requires that each of the K datasets contain the same n items, while allowing each dataset to be drawn from a different distribution and to contain different *feature* sets. For example one dataset could contain log-Normally distributed gene-expression data, whilst another contains Multinomial phenotypic attributes.

Previous implementations of this algorithm have been in the MATLAB programming language, whose resulting performance limited clustering to only a few hundred items. In this work, we improve the implementation such that tens of thousands of items may be clustered, removing any *a-priori* requirement to filter the large numbers of genes available from genomic-scale studies.

2 Methods

As in Kirk et al. (2012) we employ an approximation to the Dirichlet-process (DP) mixture model, by truncating to a finite N components. Within a single dataset, the data, x, is distributed as F and parametrised by θ, with latent cluster allocations described by $c_{1\ldots n}$:

$$x_i \mid c_i, \theta \sim F(\theta_{c_i}) \,, \quad \theta_{c_i} \sim G^{(0)} \,, \tag{1}$$

$$c_i \mid \pi \sim \text{Multinomial}(\pi_1, \ldots, \pi_N) \,, \tag{2}$$

$$\pi_1, \ldots, \pi_N \sim \text{Dirichlet}(\alpha/N, \ldots, \alpha/N) \,, \tag{3}$$

where i ranges over items, α describes the Dirichlet concentration of mixture proportions $\pi_{1\ldots N}$, and θs are drawn from base distribution $G^{(0)}$. The distributions of F and G supported remain unchanged from previous publications.

Markov-chain Monte Carlo (MCMC) sampling of the posterior is performed using a hybrid sampler consisting of Gibbs and Metropolis-Hastings steps. In genomic-scale datasets the number of items to cluster increases dramatically, and runtime becomes dominated by a Gibbs step sampling c_i, termed "conditional component allocation"—which scales as $O(nNK)$, using big-O notation. In order to reduce MC integration times, we turned to algorithmic changes and a C implementation capable of exploiting Graphical Processing Unit (GPU) based computation via CUDA (Nvidia, 2013). GPU based computation requires a large degree of parallelism, and this is naturally present when a

large number of calculations are independent of each other. Within the previous algorithm this independence is, in principle, present when evaluating dependence between datasets (Kirk et al., 2012, section 2.2). However, Kirk et al. target the following conditional distribution, as a result of which each cluster assignment is dependent on all other cluster assignments:

$$c_i \mid c, \pi \sim \text{Multinomial}(\hat{\pi}_{i,1}, \ldots, \hat{\pi}_{i,N}) \,, \tag{4}$$

$$\hat{\pi}_{i,j} \propto \pi_j \int f\big(x_i \mid x_{C_j^{-i}}, \theta\big)\, \mathrm{d}\theta \,, \tag{5}$$

where C_j^{-i} is the set of items associated with cluster j except for item i, and the weights $\hat{\pi}$ are drawn using the density function f associated with F, and normalised such that $\forall i, \sum_j \hat{\pi}_{i,j} = 1$.

We therefore introduce independence in cluster allocations by explicitly sampling each clusters' parameters, θ_j, as suggested in Suchard et al. (2010):

$$\theta_j \mid C_j \sim F(x_{C_j}) \,, \tag{6}$$

$$c_i \mid \pi, \theta \sim \text{Multinomial}(\hat{\pi}_{i,1}, \ldots, \hat{\pi}_{i,N}) \,, \tag{7}$$

$$\hat{\pi}_{i,j} \propto \pi_j f\big(x_i \mid \theta_j\big) \,. \tag{8}$$

Moving this integration into the MC sampler requires additional parameters to be sampled, however the target distribution remains unchanged and, empirically, does not appreciably alter the rate at which component allocations mix.

3 Results and Discussion

The changes noted above reduce the MCMC integration time by approximately four orders of magnitude, as shown using synthetic data in Fig. 1. The statistical models used for the different datatypes are described in the online Supplementary Material. In greater detail, this total improvement results from four changes: first, the algorithmic change to the sampler resulted in a 5-fold runtime improvement across all implementations, second, the C implementation reduced runtime by 100 times, third, a further 5-fold improvement is obtained by employing cache-friendly algorithms and datastructures, finally a 30-fold improvement is obtained by enabling the optional CUDA support.

To generate the data used in Fig. 1, cluster allocations were first drawn from a DP, with concentration parameter $\alpha = 1$ and whose base distribution is the prior of each datatype. Samples were generated from the resulting components, to produce a synthetic dataset whose analysis was timed for the original MATLAB (circles), and C implementations with and without CUDA support enabled, drawn as diamonds and

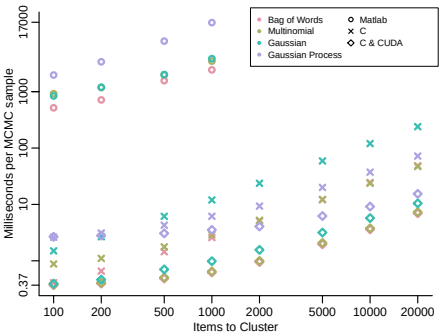

Fig. 1. Log-log plot of average runtimes for a single MC sample comparing the original to our new implementation, with and without CUDA support enabled. Runtimes are reported for a given number of items when clustering datasets of individual datatypes, with each point's datatype is indicated by its colour and the implementation by the point's style. The plateau in runtime at smaller item counts is due to the per-item runtime being dominated by other constant factors.

crosses respectively. This was repeated five times and median runtimes used for plotting. The only free parameter in MDI that significantly affects runtime is the DP truncation, N, which here is fixed at 50. Ensuring N is approximately double the number of occupied clusters provides a good trade off between runtime and accuracy. Runtime scales almost linearly with the number of items and features present in the data, whilst the number datasets scales as $O(2^{K^2})$ placing a practical upper limit on the number of datasets at six. The performance difference visible between Gaussian and other datatypes in the new implementation is largely due to evaluating transcendental functions when performing component allocation.

In a more realistic example consisting of 2000 items with a single dataset drawn from each datatype and all clustered simultaneously, the MATLAB implementation takes 37 seconds to generate a single MC sample. This time is reduced to 46 milliseconds with the C implementation, and down to 8.8 milliseconds when CUDA support is enabled. All performance evaluations were performed on 2.13GHz Intel Xeon E5606 processors with Tesla C2075 GPUs, running Ubuntu 14.10 and MATLAB 2013b, GNU GCC 4.9.1 and CUDA 6.0.1. The code has been developed and tested under recent versions of Ubuntu and OS X.

In order to obtain reasonable posterior estimates, three chains of 10^4 MC samples are generally required. Clustering three datasets of 10^3 items could occupy nine days of CPU time with the MATLAB implementation, while the CUDA implementation reduces this to less than five minutes. Alternatively, the CUDA implementation allows 10^4 items to be clustered in less than two hours, a number that would have taken months of computation time with the MATLAB implementation. This reduction in anal-

ysis time permits both significantly larger datasets to be analysed whilst also permitting a significantly faster analysis turnaround when working on smaller datasets.

Funding:
This work was supported by the EPSRC (grants EP/I036575/1 and EP/J020281/1).

References

Barash, Y. and N. Friedman (2002): "Context-specific Bayesian clustering for gene expression data." *Journal of Computational Biology*, 9, 169–91.

Kirk, P., J. E. Griffin, R. S. Savage, Z. Ghahramani, and D. L. Wild (2012): "Bayesian correlated clustering to integrate multiple datasets," *Bioinformatics (Oxford, England)*, 28, 3290–3297.

Liu, X., W. J. Jessen, S. Sivaganesan, B. J. Aronow, and M. Medvedovic (2007): "Bayesian hierarchical model for transcriptional module discovery by jointly modeling gene expression and ChIP-chip data." *BMC Bioinformatics*, 8, 283.

Liu, X., S. Sivaganesan, K. Y. Yeung, J. Guo, R. E. Bumgarner, and M. Medvedovic (2006): "Context-specific infinite mixtures for clustering gene expression profiles across diverse microarray dataset." *Bioinformatics (Oxford, England)*, 22, 1737–44.

Nvidia (2013): "Compute Unified Device Architecture," URL http://docs.nvidia.com/cuda/cuda-c-programming-guide/.

Rogers, S., A. Klami, J. Sinkkonen, M. Girolami, and S. Kaski (2009): "Infinite factorization of multiple non-parametric views," *Machine Learning*, 79, 201–226.

Savage, R. S., Z. Ghahramani, J. E. Griffin, B. J. de la Cruz, and D. L. Wild (2010): "Discovering transcriptional modules by Bayesian data integration," *Bioinformatics (Oxford, England))*, 26, i158–i167.

Savage, R. S., Z. Ghahramani, J. E. Griffin, P. Kirk, and D. L. Wild (2013): "Identifying cancer subtypes in glioblastoma by combining genomic, transcriptomic and epigenomic data," in *International Conference on Machine Learning*.

Suchard, M. A., Q. Wang, C. Chan, J. Frelinger, A. Cron, and M. West (2010): "Understanding GPU Programming for Statistical Computation: Studies in Massively Parallel Massive Mixtures." *Journal of Computational and Graphical Statistics*, 19, 419–438.

Yuan, Y., R. S. Savage, and F. Markowetz (2011): "Patient-specific data fusion defines prognostic cancer subtypes." *PLoS computational biology*, 7, e1002227.